

2-1.6 MULTIPLICATION

Multiplication increases programming complexity. In addition to the addition and subtraction instructions, the use of the shift and rotate instructions is required. The general algorithm for binary multiplication can be illustrated by a short example:

- (1) Test the least significant multiplier bit for 1 or 0.
 - (a) If it is 1, add the multiplicand to the result, then 2.
 - (b) If it is 0, then 2.
- (2) Shift the multiplicand left one bit.
- (3) Test the next more significant multiplier bit; then 1a or 1b.

DECIMAL	BINARY	
13	1101	MULTIPPLICAND
11	1011	MULTIPLIER LSB=1; ADD MULTIPPLICAND TO RESULT (A)
	1101 (A)	
13	1101 (B)	SHIFT MULTIPPLICAND LEFT ONE BIT (B)
	100111 (C)	LSB+1 = 1; ADD MULTIPPLICAND TO RESULT (C)
13	1101 (D)	SHIFT MULTIPPLICAND LEFT ONE BIT (D)
	1101 (E)	LSB+2 = 0; SHIFT MULTIPPLICAND LEFT 1 (E)
143	10001111 (F)	LSB+3 = 1; ADD MULTIPPLICAND TO RESULT (F)
	128 + 15 = 143	

Signed binary numbers in 2's complement form cannot be multiplied without correcting for the cross product terms which are introduced by the 2's complement representation of negative numbers. There is an algorithm which generates the correct 2's complement product. Since positive binary numbers are correct 2's complement notations, they also may be multiplied using this procedure. It is called Booth's Algorithm. Simply stated the algorithm says:

- (1) Test the transition of the multiplier bits from right to left assuming an imaginary 0 bit to the immediate right of the multiplier.
- (2) If the bits in question are equal, then 5.
- (3) If there is a 0 to 1 transition, the multiplicand is subtracted from the product, then 5.
- (4) If there is a 1 to 0 transition, the multiplicand is added to the product, then 5.
- (5) Shift the product right one bit with the MSBit remaining the same. (This has the same effect as shifting the multiplicand left in the previous example).
- (6) Go to 1 to test the next transition of the multiplier.

The following example (Figure 2-1.6-1) shows the typical steps involved in an actual calculation.

A Flowchart and Assembly Listing for a program using the MC6800 instruction set is shown in Figures 2-1.6-2 and 2-1.6-3, respectively. The results of simulating this program, Figure 2-1.6-4, shows worst case processing time to be approximately 1.662 msec. The worst case condition results when alternate additions and subtraction are required in each of the 16 loops required to have the result in the proper location.

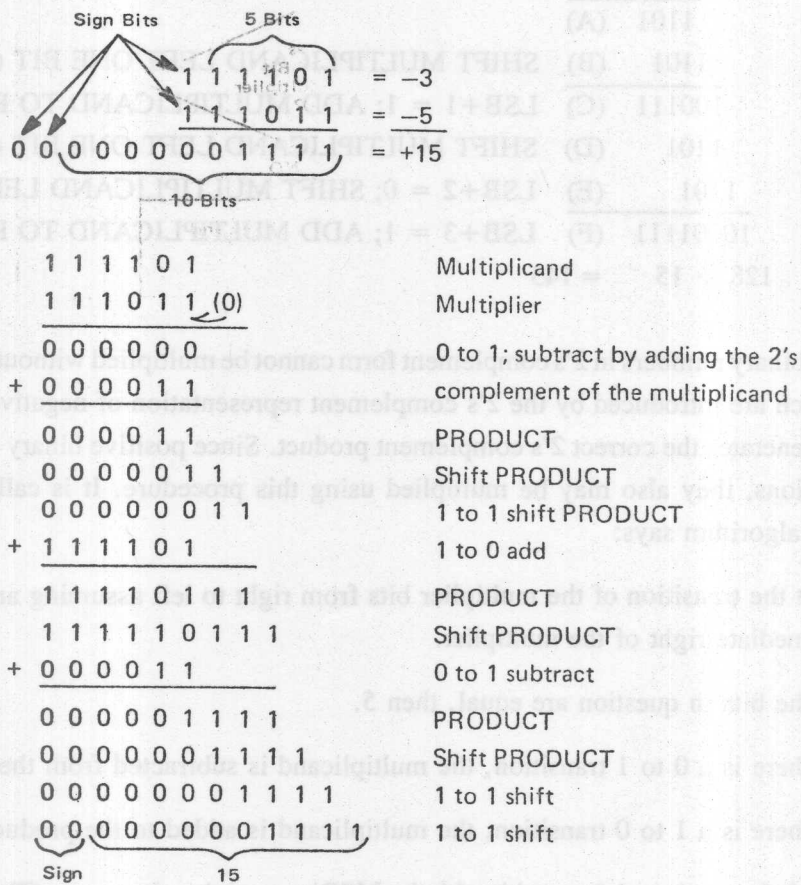


FIGURE 2-1.6-1. Multiplication Using Booth's Algorithm

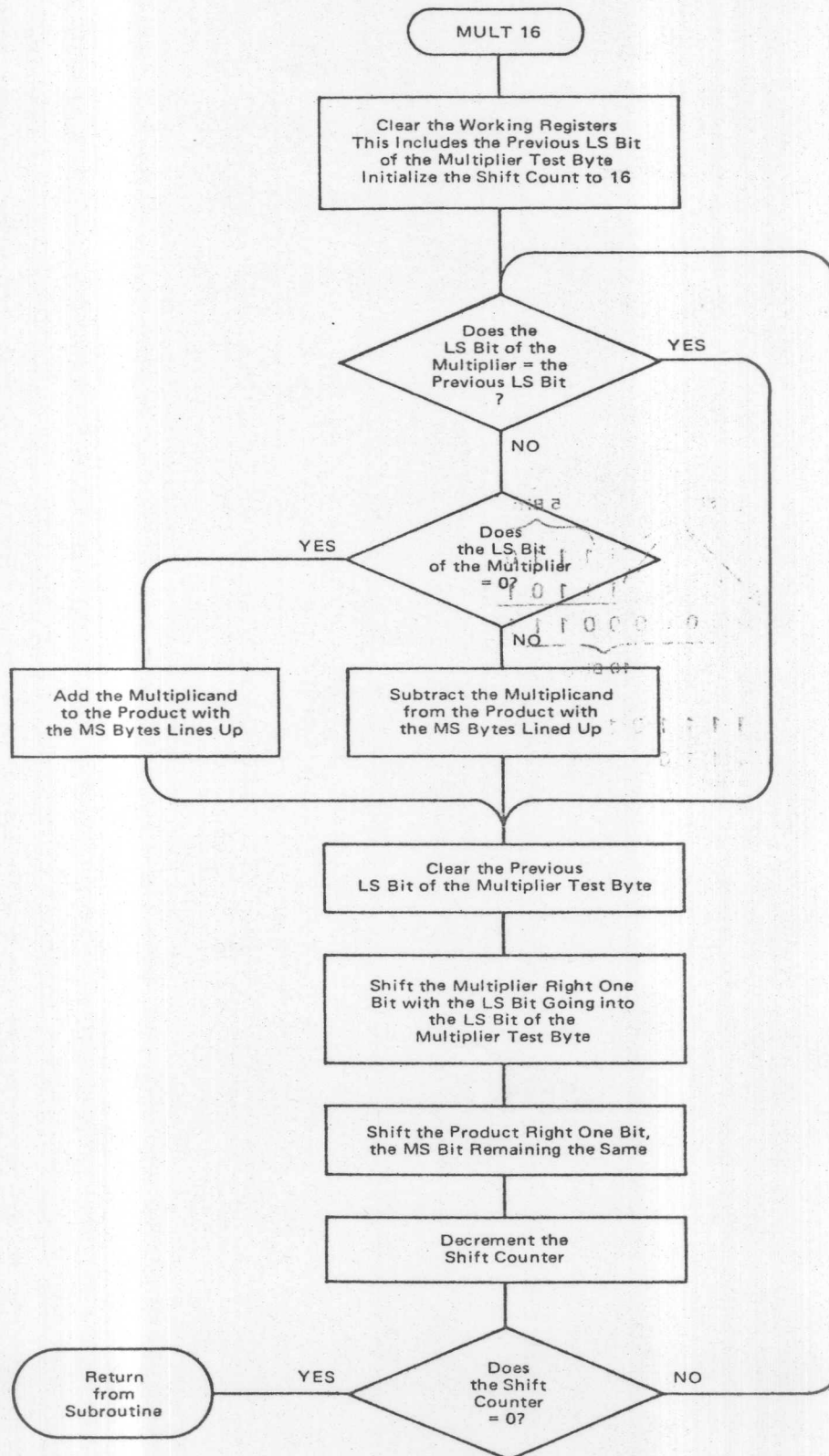


FIGURE 2-1.6-2. Flow Chart for Booth's Algorithm

